jamk.fi

GPS-seurantasovellus JavaScriptillä

Petri Mölläri

Opinnäytetyö

Huhtikuu 2015

Ohjelmistotekniikan koulutusohjelma

Tekniikan ja liikenteen ala

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Mölläri, Petri	Opinnäytetyö	12.04.2015
	Sivumäärä 39	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: x

Työn nimi
GPS-seurantasovellus JavaScriptillä

Koulutusohjelma
Ohjelmistotekniikka

Työn ohjaaja(t)
Ari Rantala

Toimeksiantaja(t)
Tmi Pekan GPS-seuranta

Tiivistelmä

Opinnäytetyössä toteutettiin GPS-seurantasovelluspalvelu erilaisia urheilutapahtumia varten. Tavoitteena oli toteuttaa erilaisille päätelaitteille hyvin skaalautuva sovellus, joka olisi täysin alustariippumaton. Erityisestä huomiota kiinnitettiin mobiililaitteiden käyttöliittymän toteutukseen ja sen käytön sujuvuuteen.

Toimeksiantajana opinnäytetyössä toimi GPS-seurantasovelluspalvelua erilaisiin urheilutapahtumiin tarjoava Tmi Pekan GPS-seuranta. Opinnäytetyössä kehitetty sovellus pohjautui asiakkaan Java-sovelmana toteutettuun ratkaisuun. Työ aloitettiin tutkimalla erilaisia vaihtoehtoja karttadatan käsittelyä varten. Viidentoista valitun ratkaisun joukosta otettiin lopulta tarkempaan tutkintaan neljä erilaista vaihtoehtoa. Näiden neljän valitun ratkaisun osalta tehtiin tarkempaa selvitystä niiden soveltuvuudesta tähän asiakasprojektiin. Tärkeimpiä kriteerejä olivat soveltuvuus mobiililaitteille, laajennettavuus ja suorituskyky. Lopulta valittiin yksi toteutus, jonka pohjalta lähdettiin toteuttamaan uutta GPS-seurantasovellusta.

Työtä lähdettiin toteuttamaan käyttämällä karttadatan käsittelyyn Leaflet-kirjastoa. Apuna käytettiin myös asiakkaan toteuttamaa pohjaa kilpailun datan hallintaan, jota käytettiin yhdessä Leaflet-kirjaston kanssa. Käyttöliittymää lähdettiin toteuttamaan kokonaan alusta asti käyttämällä apuna Leaflet- ja muita JavaScript-kirjastoja uusien web-tekniikoiden kanssa siten, että sovellukseen saadaan sisällytettyä tuki eri mobiilialustoille.

Vaikka työn valmistuminen lopulta viivästyi merkittävästi, saatiin kuitenkin aikaan toteutus, johon asiakas oli tyytyväinen ja jota asiakas pystyi lähteä kehittämään tuotantokäyttöön soveltuvaksi. Tärkeimmät vaatimukset projektin osalta saatiin täytettyä ja käyttöliittymä vastasi lopulta asiakkaan odotuksia. Sovelluksen lopullisella versiolla oli mahdollista seurata urheilukisoja suorana lähetyksenä tai uusintana jälkikäteen ja se oli käytettävissä kaikilla yleisimmin käytössä olleilla päätelaitteilla.

Avainsanat (asiasanat)
web-sovellus, web-teknologiat, GPS-seuranta, mobiilisovellus, HTML5, CSS3, JavaScript

Muut tiedot

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

Description

Author(s)	Type of publication	Date
Mölläri, Petri	Bachelor's thesis	12.04.2015
		Language of publication: Finnish
	Number of pages: 39	Permission for web publication: x

Title of publication
GPS Tracking software implemented using JavaScript

Degree programme
Software Engineering

Tutor(s)
Rantala, Ari

Assigned by
Tmi Pekan GPS-seuranta

Abstract

The thesis was assigned by Tmi Pekan GPS-seuranta that offers GPS tracking services for sport events. The project's goal was to implement a new version of tracking software that uses modern web technologies where the old version was implemented as Java applet. One of the project's key aspects was to offer good support for different mobile device platforms.

The project's first step was to research different solutions on implementation of mapping to this software. From fifteen different solutions four were selected to a closer analysis. The key properties for this project were support for mobile devices, ease to extend it and performance. At the end one solution was selected and the implementation of this software project was started by using the selected solution.

The implementation of the software was started by using Leaflet JavaScript library along with the project's client source code for data management of events. The program's user interface was implemented using Leaflet and other JavaScript libraries with the latest web technologies in order for the program to would achieve a wide support for different mobile device platforms.

Even though the project's schedule was hugely delayed the customer was satisfied with the actual outcome when the project was finally finished. Also, the most critical software requirement specifications were met and the software's user interface met the customer's expectations. The customer was able to continue with the program's further development that it would be ready for production use. In the final version of the project's software users could use it to watch live GPS data from sport events or to watch their replays.

Keywords/tags (subjects)

web application, web technologies, GPS tracking, HTML5, CSS3, JavaScript

Miscellaneous

Sisältö

Kuviot

KÄSITTEET

Ajax

Asynkronoidusti datan vaihtaminen selainohjelman ja palvelimen välillä

CSS

CSS (Cascading Style Sheets) on WWW-dokumenteille toteutettu tyyliohjeiden laji. CSS3 on viimeisin versio CSS:sta.

Google AdSense

Googlen luoma mainosohjelma verkkosivuille.

HTML

Avoimen standardin kuvauskieli tekstidokumenteille. HTML5 on viimeisin versio HTML standardista.

Java

Sun Microsystemsin kehittämä ohjelmistoalusta ja ohjelmointikieli.

Java-sovelma

Selaimessa suoritettava Java-ohjelma.

JavaScript

Komentosarjakieli, jota pääsääntöisesti käytetään Web-ympäristössä.

Liitännäinen

Komponentti, joka lisää ominaisuuksia jo olemassa olevaan alustaan.

Loppukäyttäjä

Henkilö, joka käyttää kyseistä palvelua tai sovellusta.

Lähdekoodi

Tekstimuotoinen listaus ohjelmointikielestä.

Muuttuja

Tietovarasto, josta voidaan lukea tietoa ja johon voidaan kirjoittaa tietoa.

Ohjelmointirajapinta

Ohjelmointirajapinta (Application programming interface, API) on määritelmä, jolla eri ohjelmat voivat vaihtaa tietoa keskenään

Refaktorointi

Lähdekoodin sisäisen rakenteen parantamista siten, että ohjelman alkuperäinen toiminnallisuus kuitenkin säilyy.

Regressio

Esimerkiksi aiemmin toiminut ominaisuus lakkaa toimimasta lähdekoodiin tehtyjen uusien muutosten myötä.

Single-Page Application (SPA)

SPA on yhdelle HTML-sivulle tehty sovellus.

Web

Web (World Wide Web, WWW) on Internetissä toimiva hypertekstijärjestelmä.

WebKit

Yksi verkkoselainten selainmoottoreista, jonka tehtävänä on renderöidä websivut.

WHATWG

Web Hypertext Application Technology Working Group on yhteisö, joka on mukana kehittämässä HTML:ää ja siihen liittyviä tekniikoita.

Widget

Pieni ohjelma rajoitetuilla ominaisuuksilla, joka voidaan suorittaa web-sivulla.

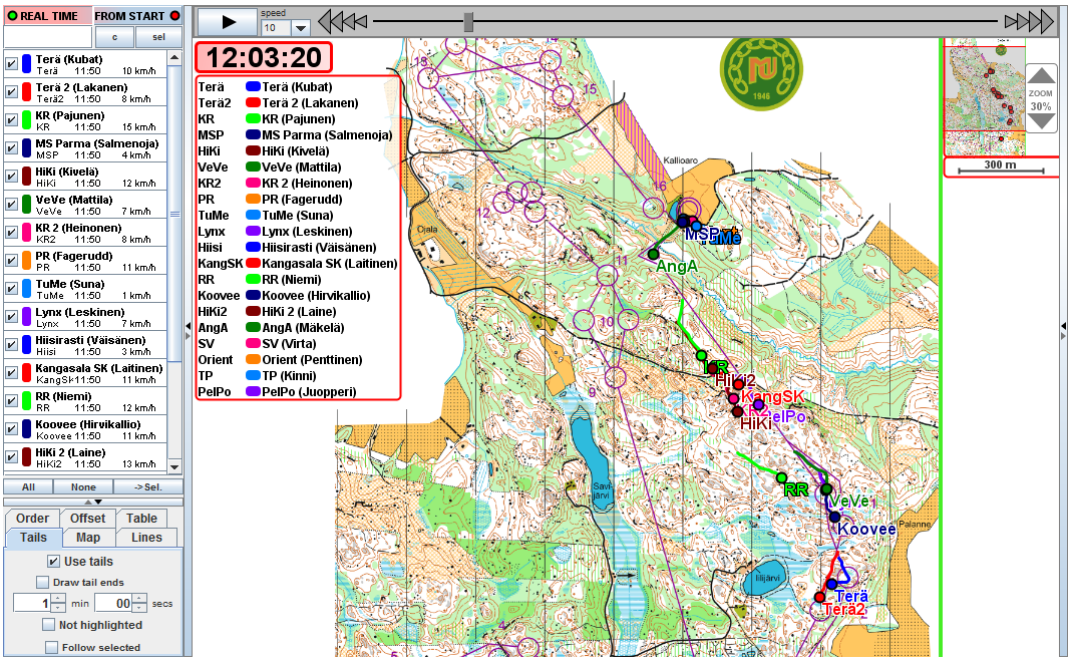
1. Työn lähtökohdat

1.1. Toimeksiantaja

Työn toimeksiantajana toimi Tmi Pekan GPS-seuranta, joka tarjoaa GPS-seurantapalvelua erilaisia urheilutapahtumia varten. Kyseisen toiminimen omaava Pekka Varis toimii myös luennoitsijana Jyväskylän ammattikorkeakoulussa.

1.2. Työn tavoitteet

Lähtökohtana työlle oli toteuttaa web-tekniikoilla uudistettu versio vanhasta olemassa olevasta Javalla tehdystä GPS-seurantasovelluksesta (ks. kuvio 1.). Työn pääkohtina olivat HTML5-, CSS3- ja JavaScript-tekniikoiden käyttö, käyttöliittymän päivittäminen nykyaikaisemmaksi, tuki mobiililaitteille ja ylläpidettävyys. Etenkin sovelluksen tuki mobiililaitteille oli olennainen osa projektia, sillä sovelluksen Java-versio ei tukenut ollenkaan ohjelman käyttöä mobiililaitteilla.



Kuvio 1. Tmi Pekan GPS-seurannan GPS-seurantasovellus (Java-sovelma).

1.3. Taustatutkimus

Asiakas toivoi käyttöliittymän osalta, että uusi versio sovelluksesta noudattaisi samanlaista logiikkaa kuin nykyinen tuotannossa oleva versio. Näin ollen taustatutkimus keskittyi vanhan sovelluksen lähempään tarkasteluun käyttöliittymän osalta. Lisäksi sovellukselle oli myös huomattava määrä erilaisia kilpailijoita, joten myös kilpailevien sovellusten tutustumiseen käytettiin aikaa. Tällä pyrittiin löytämään hyviä ideoita käyttöliittymän toteutukseen ja myös välttämään esille tulleita ongelmia kilpailijoiden toteutuksien osalta.

Tutkimus keskittyi lopulta siihen, millaisilla tekniikoilla ja menetelmillä käyttöliittymästä saataisiin tehtyä selkeä ja että sen käytettävyys olisi etenkin mobiililaitteilla mielekästä. Samalla kuitenkin sovelluksen tulisi tarjota kattavasti erilaisia ominaisuuksia ja niihin tulisi päästä helposti ja nopeasti käsiksi.

2. Tehtävän kuvaus ja vaatimukset

Projektin tavoitteena oli tehdä uusittu versio vanhasta GPS-seurantasovelluksesta käyttämällä nykyaikaisia web-tekniikoita. Tuotannossa oleva GPS-seurantapalvelu on käytössä etenkin suunnistus- ja hiihtokilpailuissa, mutta sitä voidaan käyttää myös muiden urheilutapahtumien seurantaan.

Sovelluksen nykyinen versio on toteutettu käyttämällä Java-ohjelmointikieltä, joka ei enää nykyään ole kovin suuressa suosiossa käytettynä selaimissa ajettavina Java-sovelmina. Java-sovelmien ongelmia ovat puutteellinen tuki mobiililaitteille, sovelluksen raskaus, vanhentunut tekniikka ja tietoturvaongelmat. Java-sovelmien tietoturvaongelmat näkyvät etenkin käyttäjille ikävinä tietoturva varoituksina, kun mennään ensimmäisen kerran sivustolle jossa suoritetaan Java-sovelmia.

GPS-seurantasovelluksen uusi versio haluttiin toteuttaa nykyaikaisilla web-teknologioilla. Web-sovellukset ovat viime vuosina tehneet kovasti tuloaan ja etenkin nykyisten suorituskykyisten mobiililaitteiden kohdalla pystytään helposti tekemään raskaita ja monimutkaisia web-sovelluksia, jotka ovat alustariippumattomia ja toimivat myös mobiililaitteillakin.

Projektin alussa listattiin asiakkaan kanssa kaikki esille tulleet vaatimukset.

- SPA-sovellus
- Toteutus HMTL5-, CSS3- ja JavaScript-tekniikoilla
- Vähintään samat perustoiminallisuudet tulee löytyä kuin nykyisestä Java-sovelluksesta
- Tutkimustyö karttadatan esittämiseen ja sen käyttöönottaminen projektissa
- Sovelluksen toimiminen monilla eri alustoilla
- Mobiililaitteilla toimiva käyttöliittymä
- Mobiililaitteilla sovelluksen ajaminen jouhevaa
- Ylläpidettävyys ja koodin uudelleenkäytettävyys
- Modernisoitu käyttöliittymä
- Helppo laajennettavuus

- Modulaarisuus
- MVC-arkkitehtuuri

3. Käytetyt teknologiat ja kirjastot

3.1. Kirjastot ja teknologiat käyttöliittymän toteutukseen

Vuosien aikana on web-sovelluksille ilmaantunut monia erilaisia kirjastoja joiden myötä kaikkea ei tarvitse enää itse lähteä toteuttamaan täysin alusta, vaan tarjolla on monia erilaisia hyvin toimivia ratkaisuja. Käyttöliittymän toiminnallisuuden toteuttamisen avuksi otettiin käyttöön seuraavat alla luetellut JavaScript-kirjastot:

- colpick Color Picker
- jQuery
- jQuery UI
- Spin.js.

Projektin kannalta jQuery nousi Leafletin ohella tärkeään asemaan projektin toteutuksessa. jQuery on avoimeen lähdekoodiin perustuva JavaScript-kirjasto, jonka kehittämisestä ja ylläpidosta nykyään vastaa jQuery-säätiö. Sen tarkoituksena on yksinkertaistaa HTML:n asiakaspuolen skriptaamista. Käytännössä se tarjoaa paljon valmiita metodeja erilaisten tapahtumien, animaatioiden, Ajaxin ja HTML-sivujen käsittelyyn ja manipulointiin. (jQuery n.d.a.; jQuery n.d.b.; jQuery Foundation n.d.)

jQuery UI on jQuery-säätiön kehittämä, ja se perustuu pitkälti kyseiseen JavaScript-kirjastoon. jQuery UI -kirjasto tarjoaa laajan valmiin valikoiman erilaisia toiminnallisuuksia, widgettejä ja efektejä. Tätä kirjastoa käyttäen pystytään helposti esimerkiksi lisäämään sovellukseen valikoita ja liukusäätimiä, joihin tarvitsee vain yhdistää haluttu toiminallisuus. (jQuery UI n.d.)

Spin-kirjasto tarjosi helposti käytettävät latausanimaatiot sivuille ja Color Picker mahdollisti lisäämään sivuille erittäin monipuolisen värivalitsimen, jota käytetään projektin sovelluksessa valitun kilpailijan reitin ja merkin värin vaihtamiseen. Käytännössä nämä kirjastot tarjosivat yhden ominaisuuden, jonka

sai nopeasti otettua käyttöön projektissa. (jQuery Color Picker n.d.; Spin.js n.d.)

3.2.1. Eri vaihtoehdot karttadatan näyttämiseen

Erilaisia kartan ja kilpailijoiden reittien piirtämiseen tarkoitettuja JavaScript-kirjastoja oli tarjolla lukuisia. Poikkeuksen tähän joukkoon tosin tekevät Canvas, joka on HTML5:n mukana tullut uusi elementti grafiikan piirtämistä varten ja Google Maps API, joka on JavaScript-rajapinta Googlen oman Maps palvelun käyttöönottamiseksi sovelluksiin. Alla on lista tutkimuksen aikana löytyneistä mahdollisista ratkaisuista karttadatan käsittelyyn sovelluksessa. Lista koostuu kaiken kaikkiaan viidestätoista erilaisesta ratkaisusta. (50 JavaScript Libraries and Plugins for Maps n.d.)

- AmMap
- D3
- Geo5
- GeoExt
- Google Maps API
- HTML5 Canvas
- Jump
- Kartograph
- Leaflet
- MapBox
- MapQuery
- Modest Maps
- OpenLayers
- PolyMaps
- WebGL Globe

Valintakriteereinä tarkempaan tutkimukseen valituille kirjastoille olivat kehityksen tila, helppo laajennettavuus, saatavilla olleen dokumentaation taso ja soveltuvuus tämän projektin sovelluksen käyttöön ja tuki mobiililaitteille. Listalla olleista monia vaihtoehtoja oli helppo sulkea pois jo sen vuoksi, että ne olivat vanhentuneita ja tuki ja kehitys niille oli lopetettu jo vuosia sitten. Monet näistä

vaihtoehdoista eivät myöskään soveltuneet käytettäväksi tähän projektiin, sillä niiden ominaisuudet olivat paikoin erittäinkin puutteellisia piirtämiseen liittyvien toiminnallisuuksien kannalta. Sen lisäksi monet vaihtoehdoista eivät olleet saavuttaneet suurta käyttäjäkuntaa, mikä näkyi heikkona tuen saamisena. Myös dokumentaation taso saattoi olla puutteellista ja vähäistä. (25 Useful JavaScript Libraries And Tools for Creating Interactive Maps 2014; How do various JavaScript mapping libraries compare n.d.)

Lopulta tarkempaan tutkimustyöhön päätyi neljä vaihtoehtoa: Leaflet, OpenLayers, Google Maps API ja HTML Canvas. Kaikki neljä valittua vaihtoehtoa täyttivät karsintakriteerit. Tästä huolimatta nämä vaihtoehdot erosivat tosistaan merkittävästi teknisellä puolella. Luvuissa 3.2.2 – 3.2.5 käydään tarkemmin jokainen lähempään tarkasteluun valittu toteutus läpi ja annetaan perustelut valitulle tekniselle toteutukselle karttadatan käsittelyyn.

3.2.2. OpenLayers

OpenLayers on avoimeen lähdekoodiin perustuva JavaScript-kirjasto karttadatan esittämiseen verkkoselaimissa. OpenLayersin kehitys alkoi kesäkuussa 2005 MetaCarta-nimisen yrityksen toimesta ja ensimmäinen versio siitä julkaistiin kesäkuussa 2006. Marraskuusta 2007 lähtien OpenLayers siirtyi Open Source Geospatial Foundation-järjestön projektiksi. OpenLayers oli ensimmäisen avoimeen lähdekoodiin perustuva kartoitus-API, joka saavutti suuren menestyksen. (OpenLayers n.d.)

OpenLayersin pääversio numero 2 oli jo päässyt lähelle kymmenen vuoden ikää, kun tätä tutkimusta alettiin tekemään tammikuussa 2014. Tänä aikana kirjasto on paisunut valtavan kokoiseksi, ja se tarjoaa erittäin kattavasti ominaisuuksia laidasta laitaan. OpenLayersion versionhallintaan onkin kertynyt yhteensä yli 23000 muutosta menneiden vuosien aikana. Lisäksi sen ympäriltä löytyy paljon erilaista materiaalia. Näin ollen OpenLayersia voidaan pitää erittäin pitkälle hiottuna tuotteena. (OpenLayers 2 n.d.)

Tämän kirjaston suurimpina etuina onkin saatavan dokumentaation määrä sen rajapinnan osalta ja erittäin kattava ominaisuustarjonta oletuksena muihin ver-

rattuna. Samalla tosin laaja tarjonta ominaisuuksien suhteen näkyy kirjaston massiivisena tiedostokokona (lähes 1 Mb). Mobiililaitteiden kannalta tämä ei ole paras tilanne, koska suuri kokoisten kirjastojen käyttö näkyy sovelluksen latausajoissa. Kirjaston vanha ikä näkyy myös siinä, että kirjastoa ei alunperin ollut suunniteltu mobiililaitteita ajatellen. Suorituskyky ei ole uusiin kilpailijoihin verrattuna samalla tasolla ja OpenLayers 2 ei tue yhtä suoraan mobiililaitteiden tapahtumia, kuten kosketusnäytön eri tapahtumia. Kirjaston käyttöönotto on näin ollen työläämpää mobiililaitteita ajatellen. Lisäksi käyttöliittymä on tarjolla olevien komponenttien osalta erittäin vanhentuneen näköinen ja tämä asettaa myös lisää työmäärää, sillä OpenLayersin komponenttien oletus ulkoasuun tulisi tehdä suuria muutoksia.

Vaikka OpenLayers tarjoaa kattavasti ominaisuuksia, sille on tarjolla kattavasti dokumentaatiota ja esimerkkejä ja se on vapaan lähdekoodin projekti, niin sillä on myös tiettyjä suuria haittapuolia. Koska tässä projektissa mobiililaitteet ovat tärkeässä asemassa, OpenLayers 2 ei tältä kantilta ole paras mahdollinen valinta sen tarjoaman suorituskyvyn ja käyttöönoton kannalta mobiililaitteille. Lisäksi täytyisi kiinnittää huomiota myös kirjaston komponenttien oletus ulkoasuihin, sillä projektin vaatimuksena oli käyttöliittymälle moderni ulkoasu. Nykyisellään OpenLayers 2 käyttöliittymäkomponentit ovat paikoin liiankin vanhahtavan näköisiä.

Näihin ongelmiin oli tulossa merkittäviä parannuksia OpenLayers 3 version myötä, mutta tutkimuksen aikana se oli vielä pahasti kesken. Omat haasteensa asetti myös se, kumpaa versiota käyttäen projektia lähdettäisiin toteuttamaan. Tutkimuksen aikana selvisi se, että kyseiset versiot eroavat merkittävästi toisistaan. Tämä tarkoitti käytännössä sitä, että OpeanLayers 2 versiota käyttävää sovellusta ei voi helposti päivittää käyttämään OpenLayers 3 versiota. Uudempaan versioon siirtyminen vaatisi merkittäviä muutoksia lähdekoodiin, jotta voitaisiin käyttää uutta versiota kirjastosta. OpenLayers 3 aikaisen beta-version käyttäminen taas asetti ongelmat puutteellisen dokumentaation osalta uusista ominaisuuksista ja muutoksista, sekä mahdolliset esille tulevat vakavat bugit kehitysversion myötä. (OpenLayers 3 n.d.)

3.2.3. Leaflet

OpenLayersín ja Google Maps API:n ohella Leaflet on kehittäjien keskuudessa suosituimpien JavaScript-kirjastojen joukossa. Leaflet on avoimen lähdekoodin JavaScript-kirjasto, jonka kehittäminen alkoi vuoden 2010 aikana. Leafletia voidaankin pitää OpenLayersin suorana kilpailijana. (Leaflet (software) n.d.)

Leafletin käytössä ongelmallista oli sen varhainen kehitysvaihe. Projektin aloittamisen aikana tarjolla oli Leafletista käytettäväksi vakaa 0.7.3 julkaisu ja kehityksen alla ollut 0.8 versio. Projektiin valittiin Leafletista versio 0.8, joka tarjosi etenkin suorituskyvyn kannalta merkittäviä parannuksia.

Muihin kilpailijoihin verrattuna Leafletin parhaita puolia on se, että kirjasto on kooltaan pieni, mutta silti se tarjoaa kaikki tarvittavat ominaisuudet, joita kartoitussovelluksissa tarvitaan. Tämän ohella Leafletia on tarvittaessa helppo lähteä laajentamaan omilla lisäosilla ja sille on jo valmiiksi tarjolla kattavasti erilaisia hyödyllisiä lisäosia. Näin ollen Leaflet on kevyt kirjasto ottaa käyttöön sovelluksen suorituskyvyn kannalta ja tarvittaessa sille voi kehittää itse hyvinkin paljon uusia ominaisuuksia. (Leaflet - a JavaScript library for mobile-friendly maps n.d.)

Uutena kilpailijana Leafletiä lähdettiin alusta asti kehittämään myös mobiililaitteita ajatellen. Tämä näkyy käytännössä sovelluksen kehittämisen osalta siinä, että Leaflet tukee oletuksena mobiililaitteiden erilaisia tapahtumia ilman erillisiä toimenpiteitä. Suorituskyky on myös yksi Leafletin vahvuuksista, etenkin OpenLayers 2 -kirjastoon verrattuna. Leaflet-projektissa onkin alusta asti myös kiinnitetty huomiota suorituskykyyn mobiililaitteiden osalta. Lisäksi Leaflet ja sen käyttöliittymäkomponentit ovat ulkoasultaan erittäin pelkistetyn ja modernin näköisiä, joten se jo oletuksena tukee projektin vaatimuksia modernin käyttöliittymän osalta.

Leafletin suurin ongelma lopulta on myös sen uutuus. Leafletin kehitysaste näkyykin siinä, että sillä on riesana suuri määrä erilaisia tunnettuja bugeja. Vaikka Leaflet täyttää erittäin hyvin kaikki asetetut kriteerit karttadatan näyt-

tämiseen asetetulle toteutukselle, niin suuri kysymysmerkki onkin, onko Leaflet-kirjasto tarpeeksi stabiili käyttöönotettavaksi sovellukselle tai palvelulle, jota myydään eteenpäin muille henkilöille. (Compare Projects - Open Hub 2015)

3.2.4. Google Maps API

Google Maps API on Googlen käynnistämä projekti, joka sai alkunsa kesäkuussa 2005. Tarkoituksena oli tarjota web-kehittäjille mahdollisuus Google Mapsin integroimiseen omille web-sivuille. Google Maps API onkin tällä hetkellä yksi eniten käytetyistä web-sovellus-rajapinnoista. (Google Maps n.d.)

Toisin kuin muut projektiin käytettäväksi valitut teknologiat, Maps API ei perustu vapaaseen lähdekoodiin, joten sen lähdekoodiin ei ole mahdollista päästä käsiksi ja tutkia lähdekoodi tasolla sovelluksen teknistä toteutusta ja siihen ei näin ollen ole mahdollista päästä tekemään muutoksia. Tämä tuo mukanaan omia ongelmia. Suurin haitta tässä on se, että sovellukseen ei juuri itse pääse vaikuttamaan millään tapaa. Jos API:n kanssa esiintyy ongelmia, niitä ei voi itse lähteä korjaamaan, vaan täytyy odottaa Googlen puolelta korjausta. Toisaalta taas etuna on se, että Maps API:n kehittämisestä ja ylläpidosta vastaa suuri yhtiö. (Google Developers n.d.; Google API Tutorial n.d.)

Maps API palvelulta voidaan odottaa hyvää laatua ja että se täyttää tietyt suorituskykyvaatimukset, sillä kyseinen API on erittäin laajasti käytössä erilaisissa sovelluksilla ja se toimii kaikilla mahdollisilla alustoilla. Lisäksi Googlella on isona yrityksenä riittävästi resursseja tarjota laadukas ja viimeistelty tuote, etenkin kun Googlen työntekijöitä pidetään yleisesti erittäin taitavina ja lahjakkaina. Näin voidaankin olettaa, että mahdolliset tuotantoversioon päätyneet bugit korjataan erittäin nopeasti. (Google Developers n.d.; Google API Tutorial n.d.)

Leafletin tapaan myös Google on pyrkinyt tarjoamaan mahdollisimman hyvin toimivan ratkaisun mobiililaitteita ajatellen ja mobiililaitteet on otettu erityisesti huomioon viimeisimmän API version kehityksen aikana. Tämä käytännössä tarkoittaa sitä, että Maps API:n käyttöönottaminen mobiililaitteille on yhtä helppoa kuin Leafletin kanssa ja se tukee suoraan erityisesti mobiililaitteiden

kosketustapahtumia näytöllä. Siinä missä Leaflet ja OpenLayers ovat vapaaseen lähdekoodiin perustuvia ja mahdollistavat asioiden tekemisen monella eri tavalla, Maps API:n kanssa näin ei ole. Käytännössä API:n käyttäminen pakottaa tekemään asiat Googlen haluamalla tavalla ja näihin asioihin ei itse voi juuri mitenkään vaikuttaa. (Testing web map APIs - Google vs OpenLayers vs Leaflet 2014)

Lisenssiehdot asettavat myös omat ongelmansa sovelluksen suhteen. Käytännössä Googlen lisenssiehdoissa on mainittu, että Google voi tuoda mainokset pakollisena osaksi Maps API:a. Toisin sanoen sovelluksen näkymään voi yllättäen alkaa ilmestymään Google AdSensen mainoksia. Riskinä on myös se, että palvelu saatetaan lopettaa kokonaan. Jos Google ei päätä tällaisessa tapauksessa laittamaan lähdekoodia jakoon, tällöin käytännössä joudutaan kiireellä toteuttamaan uusi ratkaisu sovellukselle karttadatan käsittelyä varten. Tällainen toisen tahon ylläpitämä palvelu asettaa monia riskiä ja ongelmia, jos tulee tarve saada uusia ominaisuuksia tai palvelusta itsestään löytyy kriittisiä ongelmia tuotantovaiheessa. Lisäksi jos sovellus saavuttaa tietyt rajat datansiirron suhteen, tulee Maps API palvelua käyttävän tahon ostaa lisenssi, joka tarjoaa rajattoman datamäärän siirron. Tosin nämä datamäärät koskevat vain Googlen tarjoamia karttoja ja niihin liittyviä toimintoja. (Google Developers n.d.; Google API Tutorial n.d.)

3.2.5. Oma toteutus käyttämällä HTML5 Canvasia

HTML5 Canvas poikkeaa merkittävästi muista ratkaisuista sen suhteen, että se ei varsinaisesti ole tarkoitettu suoraan karttadatan esittämiseen eikä se ole erillinen JavaScript-kirjasto, vaan se on HTML5-merkintäkieleen lisätty elementti grafiikan piirtämiseen skriptaamalla. Alkunsa Canvas sai Applen toimesta ja se esiteltiin vuonna 2004 osana Mac OS X:n WebKit komponenttia. Myöhemmin se standardisoitiin WHATWG-yhteisön toimesta osana seuraavan sukupolven web-teknologioita. (Canvas element n.d.; WHATWG n.d.; Web Hypertext Application Technology Working Group n.d.)

Muihin verrattuna Canvaksen käyttöönotto vaatii suuren määrän työtä ja myös opettelua. Käytännössä Canvas ei ole mikään valmis ratkaisu, joka voidaan

ottaa suoraviivaisesti käyttöön ja alkaa saman tien työskentelemään valmiin pohjan päällä. Se vaatii oman ratkaisun suunnittelemista ja toteuttamista alusta asti. Tämä lisää huomattavasti työmäärää ja voi jopa merkittävästi viivästyttää projektin aikataulua, jos toteutuksen kanssa törmätään suurempiin ongelmiin. (Canvas – HTML n.d.)

Omalla toteutuksella Canvasin kanssa on mahdollista saada aikaan parhaimmillaan erittäin hyvin optimoitu ratkaisu suorituskyvyn kannalta. Tämä tosin vaatii jo jonkin verran aiempaa kokemusta vastaavanlaisten projektien tekemisestä Canvasia käyttäen. Muihin valmiisiin JavaScript-kirjastoihin verrattuna tällä tuskin saavutettaisiin suurempaa suorituskykyä, vaikka onnistuttaisiinkin tekemään teknisesti erittäin onnistunut ratkaisu Canvasia käyttäen. Etuna ja haittana on myös näin ollen se, että toteutukseen pääsee vaikuttamaan erittäin paljon, jolloin voidaan tehdä juuri vain ne ominaisuudet, jotka sovellukselle tarvitaan. Toisaalta taas oman ratkaisun käyttö tällä sovelluksella vaatisi suurta työtä ylläpidon kannalta. Muilla vastaavilla vapaaseen lähdekoodiin perustuvilla valmiilla ratkaisuilla on takana aktiivinen yhteisö ja kehittäjätiimi, jolloin kaikki vastuu ei ole täysin aivan itsellä mahdollisten ongelmatilanteiden sattuessa kohdalle.

Canvasilla oman toteutuksen tekemiseen kuuluu myös suuri riski sen suhteen, että toteutuksesta saattaa tulla teknisesti erittäin huono. Etenkin jos ei ole aiempaa osaamista Canvasin käyttämisestä. Tämän myötä lopullisesta sovelluksesta saattaa tulla muun muassa suorituskyvyn suhteen erittäin raskas ja johtaa siihen, että sovellusta ei ole mahdollista käyttää mobiililaitteilla siten, että saavutettaisiin kelvollinen käyttömukavuus. Pahimmillaan projekti joudutaan keskeyttämään kokonaan, kun sovelluksen minimivaatimuksia ei pystytä täyttämään. Toinen mahdollinen vaihtoehto olisi Canvasista luopuminen ja ottaa tilalle käyttöön joku aiemmista tässä tutkimuksessa esillä olleista erillisistä kirjastoista karttadatan käsittelyyn ja esittämiseen.

Canvas vaatii näin ollen paljon työtä, että projektin lopussa saataisiin mahdolliseen tuotantokäyttöön soveltuva tuote. Lisäksi se asettaa suuria riskejä projektin läpiviemisen kannalta. Canvasilla toteutettavaa omaa ratkaisua voi-

daankin pitää projektin valmistumisen ja onnistumisen kannalta erittäin riskialttiina valintana.

3.2.6. Projektissa käyttöön otettu ratkaisu

Tarkemman tutkimuksen jälkeen päädyttiin valitsemaan projektissa käytettäväksi Leaflet JavaScript-kirjastoa karttadatan esittämiseen. Kilpailijoihinsa verrattuna siitä ei löydetty juurikaan suuria huonoja puolia ja sen käyttöönottaminen kuului riskittömämpiin vaihtoehtoihin. Kaiken kaikkiaan ominaisuuksiensa puolesta se oli vahvasti soveltuvin valinta otettavaksi käyttöön tässä sovelluksessa. Leaflet täytti juuri ne vaatimukset mitä vaadittiin kartan ja karttadatan käsittelyä ja esittämistä ajatellen.

Leafletin suurimpina etuina muihin ratkaisuihin verrattuna on sen helppo laajennettavuus ja se, että sille on valmiiksi jo tarjolla runsaasti erilaisia liitännäisiä. Sovellukseen on näin ollen helppoa ja nopeaa tuoda uusia ominaisuuksia tarpeen mukaan. Esimerkiksi Leafletille löytyy valmis liitännäinen, jolla on mahdollista piirtää kartalle erilaisia kuvioita ja viivoja ja lisätä pisteitä. Sovelluksen kannalta tälle liitännäiselle voisi löytyä hyödyllistä käyttöä pienen jatkokehityksen kautta.

Muita etuja on, että se on uusi tulokas joka on jo lyhyessä ajassa ehtinyt saavuttamaan suurta suosiota ja suuren kehittäjäyhteisön taakseen. Mobiililaitteiden puolesta sen kehityksessä on alusta alkaen myös huomioitu erittäin hyvin tuki mobiililaitteille. Käytännössä Leaflet tukee suoraan erittäin hyvin mobiililaitteita ja tukee hyvin kosketuseleitä, se ei vaadi mitään erikoistoimenpiteitä toimiakseen mobiililaitteilla.

Käytännössä lopullinen valinta tehtiin pitkälti Leafletin ja OpenLayersin välillä. OpenLayers 2 versio päätettiin hylätä sen huonon suorituskyvyn ja mobiilituen puuttellisuuden takia. OpenLayers 3 versiossa nämä asiat oli korjattu, mutta siitä oli vielä tarjolla vasta aikainen kehitysversio ja sen dokumentaatio oli puutteellista. Kahden kehitysversion väliltä päädyttiin lopulta Leaflet-kirjaston 0.8 versioon. Leafletille oli tarpeeksi hyvin dokumentaatiota tarjolla ja se täytti erittäin hyvin kaikki karttadatan käsittelyn toteutukselle asetetut kriteerit. Eten-

kin Leafletin helppo laajennettavuus oli ratkaiseva tekijä mobiililaitteiden erinomaisen tuen ja suorituskyvyn ohella. (Testing web map APIs - Google vs OpenLayers vs Leaflet 2014)

3.3. MVC-malli

Alun perin vaatimuksena oli toteuttaa sovellus käyttämällä MVC-arkkitehtuurityyliä, jonka ideana on erottaa käyttöliittymä sovellusaluetiedosta. MVC-mallin toteuttamisen avuksi sovittiin otettavan käytettäväksi Backbone JavaScript-kirjasto, jota on käytetty hyvin laajasti erilaisissa olemassa olevissa palveluissa MCV-mallin toteuttamiseksi.

MVC on lyhenne englannin kielisistä sanoista model-view-controller. Se on ohjelmistoarkkitehtuurityyli ja sen ajatuksena on erottaa ohjelman logiikka käyttöliittymästä. Kuten koko nimestä voi päätellä, MVC-malli koostuu malleista, näkymistä ja kontrollereista. Näin ollen ohjelmakoodi on jaettu kolmeen osaan. Näkymät määrittävät käyttöliittymän ulkoasun ja tietojen esittämisen käyttäjälle. Mallit vastaavat tiedon tallentamisesta, käsittelystä ja ylläpidosta järjestelmässä. Kontrolleri ottaa vastaan käyttäjältä tulevat komennot ja liittää mallit ja näkymät yhteen. (MVC-arkkitehtuuri n.d.; MVC Architecture n.d.; MVC- Malli, peruskauraa frameworkkien käyttäjille 2010)

MVC-mallin merkittävä hyöty on, että lähdekoodia voidaan helposti hyödyntää muissa vastaavanlaisissa projekteissa ja tehdä käyttöliittymään tai ohjelman logiikkaan isompiakin muutoksia niin, että koko ohjelmaa ei tarvitse lähteä toteuttamaan tyhjästä uudestaan. Tämän projektin kannalta myös yksi merkittävä MVC-mallin tuoma etu on se, että samalle logiikalle on helppo toteuttaa useita erilaisia käyttöliittymiä. Lisäksi tällä arkkitehtuurityylillä sovelluksia on helpompi laajentaa uusilla ominaisuuksilla. (MVC- Malli, peruskauraa frameworkkien käyttäjille 2010)

Vaikka MVC-mallin käyttö tarjosi merkittäviä etuja, päädyttiin tästä vaatimuksesta asiakkaan kanssa luopumaan lopulta projektin edetessä, koska todettiin sen olevan vaikeasti käytettävissä tämän tyyppisissä sovelluksissa. MVC-mallin käyttämisestä ja sen käyttöönottamisesta ei löytynyt tarpeeksi koke-

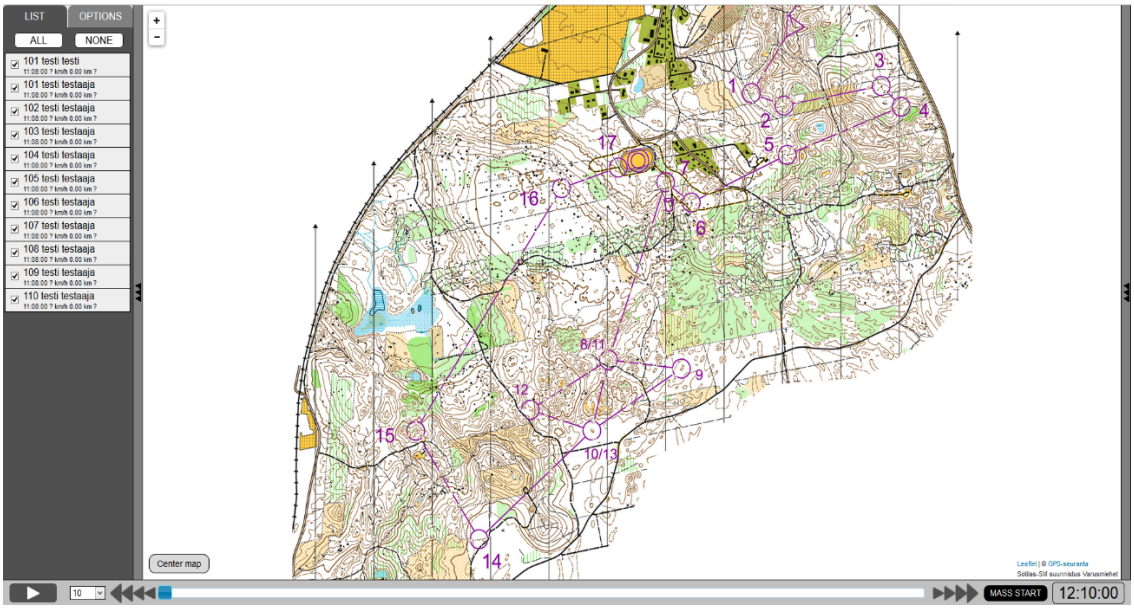
musta. Vastaavanlaisten sovellusten toteuttamisesta ei löytynyt juuri ollenkaan materiaalia ja Backbone-kirjaston dokumentaatio ei tarjonnut tarpeeksi apua, jotta MVC-mallin käyttöön ottaminen olisi ollut järkevää projektin etenemisen ja valmistumisen kannalta. (Backbone.js n.d.; Backbone.js 2014)

4. Toteutus

4.1. Käyttöliittymän suunnittelun lähtökohdat

Uuden sovelluksen käyttöliittymästä haluttiin logiikaltaan samankaltainen kuin aiemmin Javalla toteutetusta versiosta, mutta nykyaikaistetumman näköinen. Lisäksi kriteerinä oli tehdä käyttöliittymä mobiililaitteillakin toimivaksi. Ensimmäinen kuukausi toteutuksen aloittamisen jälkeen meni käyttöliittymän prototyypin tekemisessä (ks. kuvio 2.). Ohjelman varsinaista logiikkaa ei vielä lähdetty tekemään.

Käyttöliittymän suunnittelun lähtökohtina noudatettiin samanlaista asetelmaa kuin muissakin vastaavissa toteutuksissa. Sovelluksen vasemmalle laidalle toteutettiin päävalikko, joka on koko ajan näkyvillä ja josta pääsee käsiksi asetuksiin ja kilpailijalistaan. Alhaalla oli sivun levyinen osio, joka on varattu toistoasetuksille ja siihen liittyvän informaation näyttämiseen. Karttaelementti taas vie loogisesti suurimman osan näyttöalasta, jotta kilpailuja pystyisi seuraamaan mahdollisimman vaivattomasti. Lisäksi oikeaan laitaan haluttiin toinen valikko, joka on oletuksena piilotettuna.



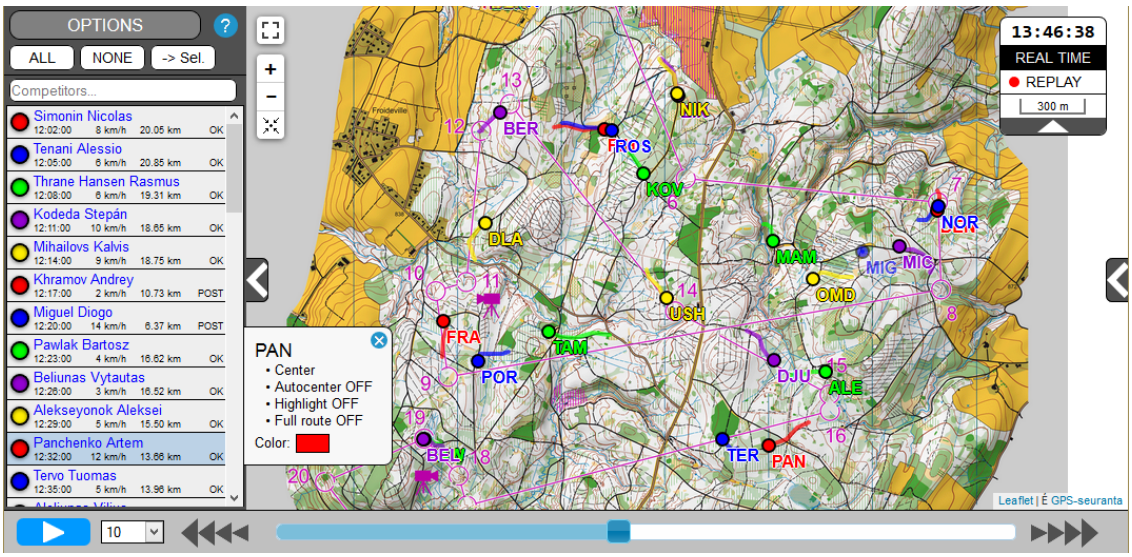
Kuvio 2. Ensimmäisiä prototyyppejä UI:sta

Etenkin mobiilikäyttöliittymä koki suuria muutoksia, sillä alunperin visiona oli toteuttaa käyttöliittymälle ulkoasu, joka olisi täysin samanlainen eri versioiden

kesken. Lisäksi projektin edetessä tuli lukuisia muutostoiveita asiakkaan suunnalta etenkin mobiiliversion käyttöliittymän suhteen.

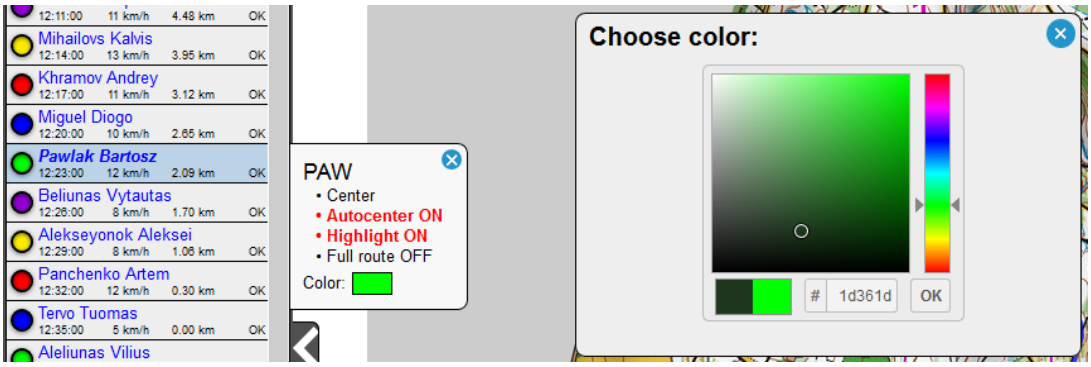
4.2. Käyttöliittymän toteutus

Käyttöliittymää lähdettiin toteuttamaan siitä näkökulmasta, että niin mobiili- kuin työpöytäversio olisivat hyvin pitkälti ulkoasunsa puolesta samannäköisiä ja toimisivat samalla logiikalla. Tästä toteutuksesta jouduttiin kuitenkin luopumaan pian erilaisten ongelmien vuoksi. Suurimmaksi ongelmaksi muodostui se, että mobiililaitteiden pienillä näytöillä ja niiden alhaisilla resoluutioilla ei voinut näyttää kovin isokokoisia elementtejä tai että erilaisia elementtejä olisi kerralla näkyvillä useita. Esille tulleiden rajoitteiden myötä päätettiin sovellukselle tehdä erikseen työpöytä- ja mobiilitilat (ks. kuvio 3.). Mobiilitila on pelkistytympi versio ulkoasultaan työpöytätilasta ja optimoitu toimimaan mobiililaitteiden alhaisemmilla resoluutioilla.



Kuvio 3. Käyttöliittymän työpöytätila.

Käyttöliittymän toiminnallisuuksien logiikan toteuttaminen oli hyvin suoraviivaista ja helppoa. Käytännössä sen suhteen ei suurempiin ongelmiin törmätty. Ainoastaan yksittäisen kilpailijan asetusvalikon ja siihen liittyvien asetusten toteuttaminen tuotti alkuun ongelmia sen suhteen, miten kyseiset ominaisuudet olisi järkevä toteuttaa (ks. kuvio 4.). Kun tähän onnistuttiin löytämään järkevän tuntuinen suunnitelma, itse logiikan tekeminen oli suhteellisen helppoa.

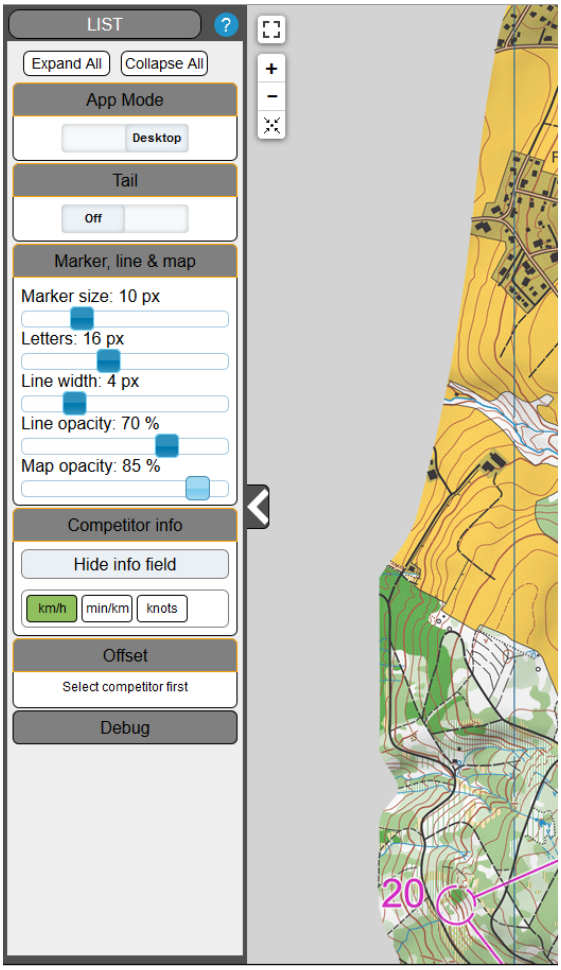


Kuvio 4. Yksittäiseen kilpailijaan liittyvät asetukset

Suurin haaste oli miettiä, miten yksittäinen HTML elementtipohja saadaan toimimaan kaikkien kilpailijoiden kesken, kuinka yksittäiselle kilpailijalle valitut asetukset tallennetaan ja kuinka kilpailijan valikko saadaan aina avautumaan oikealle kohtaa sivua. Ongelma oli etenkin, että työpöytäselainten ja mobiiliselainten välillä ikkunalle oikeiden koordinaattien laskeminen oli aluksi hankalaa. Monesti esiintyikin ongelma sen suhteen, että toisella versiolla kilpailijan valikko piirtyi väärään kohtaan verrattuna siihen, missä kohti listaa klikattu kilpailija oli.

Asetusikkunan koordinaattien asettaminen pystyakselilla saatiin lopulta tehtyä toimivaksi monien vaiheiden kautta. Käytännössä aina kun kilpailijaa klikataan listalta, haetaan kilpailijan elementille tallennettu uniikki ID numero muistiin, jolla kyseisen kilpailija tunnistetaan. Tämän jälkeen haetaan kyseisen kilpailijan elementin sijainti pystyakselissa dokumentissa, koko kilpailijalistan korkeus, asetusikkunan korkeus ja lopuksi selainikkunan korkeus. Näiden tietojen avulla sovelluksen tilasta riippuen (onko kyseessä mobiili- vai työpöytätila) vertaillaan saatuja arvoja ja niiden perusteella päätetään, tuleeko asetusikkuna esittää alhaalta ylöspäin vai ylhäältä alaspäin näkyvänä sovellusnäkymässä olevan tilan mukaan. Tämän jälkeen lasketaan oikeat koordinaatit pystyakselilla, mistä ikkunan piirtäminen tulee aloittaa. Jos asetusikkunan korkeus on pienempi kuin mitä valitun kilpailijan alapuolella on vapaata tilaa pystyakselissa, aloitetaan asetusikkunan piirtäminen samalta kohdin, missä valitun kilpailijaelementin yläreuna on. Jos taas asetusikkuna on taas liian korkea piirrettäväksi alaspäin, lasketaan piirtämisen aloittamiselle sellaiset koordinaatit, että asetusikkunan alareuna tulee samalle tasolle valitun kilpailijaelementin alareunan kanssa. Esimerkiksi kuviossa neljä on valittu kilpailija nimeltä Pawlak

Bartosz. Koska sivun alapuolella on enemmän tilaa pystyakselissa verrattuna pelkän asetusikkunan korkeuteen, on asetusikkunan piirtäminen aloitettu samalta korkeudelta kuin missä valitun kilpailijan elementti sijaitsee.



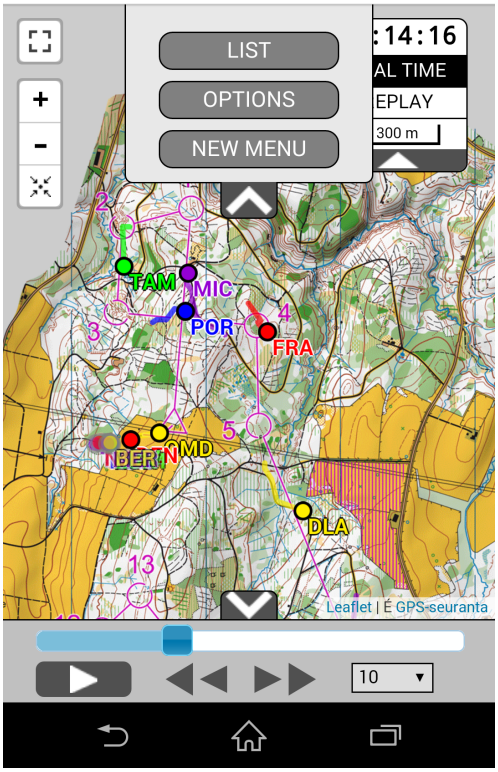
Kuvio 5. Sovelluksen asetusvalikon näkymä

Kahden eri version tila toteutettiin tekemällä kaksi erillistä CSS-tyylitiedostoa, joissa oli määritelty elementtien ominaisuuksille eri asetukset sovelluksen käyttöliittymätilan mukaan. Ylläpidettävyyden ja kehittämisen kannalta tämä ei välttämättä ollut optimiratkaisu, sillä tällaisessa toteutuksessa sai olla tarkkana, että lisäykset ja muutokset tuli tehtyä samalla tavoin kumpaankin tiedostoon. Tämä aiheuttikin useasti tilanteita, joissa sovelluksen toisen tilan käyttöliittymän ulkoasu oli rikki, koska muutoksia ei ollut tehty kumpaankin tiedostoon. Käytännössä ainoa elementti, jonka ulkoasu oli täysin sama sovelluksen eri tilojen välillä, oli sovelluksen asetusvalikko (ks. kuvio 5.).

Useat muutostoiveet mobiilitilan ulkoasun suhteen toivat omat haasteensa. Hyvinkin pieniltä kuulostaneet muutokset aiheuttivat, että lähdekoodiin sai tehdä suuriakin muutoksia haluttujen ominaisuuksien aikaan saamiseksi. Etenkin mobiilipuolella monien käyttöliittymäelementtien piilottamismahdollisuuden toteuttaminen muodostui haasteelliseksi. Lopulta muutamille käyttöliittymäelementeille piti lisätä erikseen mobiili- ja työpöytäversiot, jotta asiakkaan muutostoiveet ulkoasun suhteen saatiin toteutettua.

Lopulta sovelluksen tilan hallinnasta muodostui vaikeasti hallittava kokonaisuus. Sovelluksen tilaa vaihtaessa monien elementtien CSS tyyliasetuksiin täytyy tehdä muutoksia jQueryä käyttäen. Lisäksi omat haasteensa tähän asetti sovelluksen live- ja uusintatilat kilpailuille. Kisan ja sovelluksen tilan mukaan näytettiin tai piilotettiin tiettyjä elementtejä ja samalla monille näkyville elementeille tehtiin useita eri tyylimuutoksia lennossa. Käyttöliittymämuutosten myötä vastaan tuli usein tilanteita, joissa tarpeeksi eri tilojen välillä vaihtaminen aiheutti sen, että jotkut elementit eivät näkyneet halutulla tavalla. Tämä johti joko siihen, että käyttöliittymä meni niin pahasti rikki, että sovellusta ei enää pystynyt täysin käyttämään tai että käyttöliittymän ulkoasu näytti loppukäyttäjän näkökulmasta tökeröltä. Vaikka elementtien tyyliasetusten muuttaminen oli erittäin helppo operaatio jQueryä käyttäen, ongelmaksi muodostui kaikkien elementtien ja niiden asetusten arvojen hallinta eri tilojen välillä vaihdettaessa.

Suurimpia ongelmia tuottivat yllättäen myös kilpailijoiden valinta-painikkeet, kilpailijoiden tiedon esittäminen nimen alla ja tiettyjen mediakontrollereiden toteuttaminen. Näistä jälkimmäinen saatiin lopulta helposti toteutettua ottamalla käyttöön jQuery UI -kirjasto, joka tarjoaa monenlaisia valmiita widgettejä käytettäväksi.



Kuvio 6. Käyttöliittymän mobiilitila.

Projektin loppuvaiheessa suurimmaksi eroksi näiden versioiden välillä muodostui se, että mobiilitilassa piti pystyä mahdollisimman hyvin piilottamaan eri käyttöliittymäelementtejä, jotta mobiililaitteen ruudulla näkyisi mahdollisimman paljon itse kartasta ja siinä esitettävästä datasta (ks. kuvio 6.). Silti mobiilipuolella on tarjolla täysin samat ominaisuudet käytettäväksi kuin työpyötäversiossakin.

4.3. Karttadatan käsittely ja näyttäminen sivulla

Asiakas oli itse jo ennättänyt tekemään kokeiluja karttadatan käsittelyn osalta JavaScriptillä ja HTML Canvasilla. Projektin edetessä päädyttiin lopulta käyttämään asiakkaan valmiina olevaa pohjaa yhdessä Leaflet-kirjaston kanssa ja tätä pohjaa vasten lähdettiin kehittämään sovellusta eteenpäin. Asiakas lähti viemään eteenpäin tekemäänsä JavaScript-toteutusta kilpailijoiden datan käsittelemiseen ja esittämiseen kartalla. Koko sovelluksen toteutuksessa haluttiin asiakkaan toiveesta pitää erillään käyttöliittymä ja siihen liittyvä logiikka ja kilpailujen datan käsittelyyn liittyvä logiikka erillään toisistaan, jotta tätä asiak-

kaan puolelta tulevaa toteutusta kilpailujen datan käsittelyn osalta voitaisiin myös helposti käyttää muissakin projekteissa.

Suurimmat haasteet oli yhdistää juurikin käyttöliittymän logiikka kilpailujen datan käsittelyyn liittyvään logiikkaan siten, että kummatkin pysyisivät itsenäisinä kokonaisuuksina, mutta pystyvät keskenään välittämään dataa toisilleen. Kumpaankin alueeseen jouduttiin tekemään tiettyjä muutoksia, että nämä erillään olevat osa-alueet saatiin kommunikoimaan keskenään. Käytännössä tämä tarkoitti sitä, että tietyn käyttöliittymäkomponentin aktivoiminen lähetti kutsun käynnistää funktio dataan liittyvän logiikan puolelta. Tämä kyseinen funktio taas palautti dataa käyttöliittymälogiikan puolelle, joka sitten hoiti saadun datan esittämisen käyttäjälle oikein. Näin käytännössä kummatkin osat pysyivät itsenäisinä ja tämä mahdollisti sen, että kumpaakin osaa pystyttäisiin käyttämään muissa projekteissa tai mahdollisesti korvaamaan projektin sovelluksen toinen osa kokonaan uudella toteutuksella.

Kartan ja karttadatan piirtämiseen päädyttiin lopulta käyttämään Leaflet JavaScript-kirjastoa. Itse Leafletin käyttöönotto ja karttatiedoston esittäminen sovelluksessa oli erittäin suoraviivainen ja helppo prosessi. Käytännössä lähdekoodiin tarvitsi lisätä muutama rivi JavaScript-koodia, jolla alustettiin Leafletin käyttöönottaminen, lisättiin HTML-tiedostoon elementti johon Leaflet tekee kartan piirtämisen ja lopuksi annettiin tarkemmat tiedot kartan tyypistä ja mistä osoitteesta se haetaan.

Leafletin ominaisuuksia laajennettiin ottamalla käyttöön muutamia valmiina olleita liitännäisiä. Sovelluksen kannalta hyödyllisiä olivat valmiit laajennukset kokosivun tilaa varten ja omien painikkeiden lisäämistä varten. Projektin aikana asiakas toivoi hiiren painikkeella avautuvaa valikkoa kartan päälle. Tällekin löytyi Leafletille tehty valmis laajennus, joten kyseinen ominaisuus saatiin lisättyä erittäin nopeasti sovellukseen.

4.4. Viimeistely, testaaminen ja virheiden korjaaminen

Työn loppupuolella projektin tehtävät keskittyivät pitkälti ohjelmavirheiden korjaamiseen ja koodin siistimiseen. Sovelluksen toteuttamisen kannalta haasta-

vaa oli saada se toimimaan monilla erilaisilla mobiilipuolen alustoilla. Erityisesti sen takia, että mobiililaitteet eivät tarjoa huikeaa suorituskykyä JavaScriptin suorittamiseen ja lisäksi eri alustat ja niille tarjolla olevat selainversiot suoriutuvat erittäin vaihtelevasti web-teknologioiden parissa.

Sovelluksen testaamisen ja virheiden korjaamisen kannalta ongelmallista oli vähäinen pääsy käsiksi eri alustoihin ja laitteisiin. Monet ongelmat olivat sovelluksen toiminnan kannalta keskittyneet tiettyihin alustoihin ja niiden eri versioihin. Esimerkiksi tietyt mobiililaitteiden käyttöjärjestelmäversiot ja niissä käytössä olleet selaimet eivät välttämättä tukeneet kaikkia CSS3:n uusia ominaisuuksia.

Etenkin Applen tarjoamalla Safari selaimella esiintyi yllättäviä bugeja niin mobiili- kuin työpöytäversiolla. Toisaalta nämä ongelmat oli helppo korjata ja ne eivät olleet kovin kriittisiä sovelluksen toiminnallisuuden kannalta. Sen sijaan Windows mobiilialustoilla esiintyi vakavampia ongelmia ja niitä oli vaikeampi lähteä korjaamaan. Monet näistä ongelmista liittyivät käytössä oleviin kirjastoihin. Toisaalta taas ongelmat liittyivät enemmän Windows Phonessa olevaan Internet Exploreriin, jolla on suuria ongelmia vieläkin käytössä olevien webstandardien kanssa. Esimerkiksi Windows Phonen viimeisin 8.1 päivitys korjaa Leafletissä ilmenneen zoomaus-ongelman karttojen kanssa.

Pään vaivaa aiheutti projektin loppupuolella myös pieniltä vaikuttaneet muutokset sovellukseen, jotka lopulta saattoivat aiheuttaa uusia bugeja tai regressiota sovelluksessa.

Kun suurimmat virheet oli saatu korjattua ja asiakkaan puolelta tulleet oleelliset muutostoiveet sovelluksen kannalta oli tehty valmiiksi, projekti voitiin viimein päättää. Sen jälkeen vastuu sovelluksen jatkosta siirtyi kokonaan asiakkaalle. Tosin mahdollisissa suuremmissa ongelmatilanteissa sovelluksen suhteen luvattiin antaa tukea, jos sellaiselle on tarvetta.

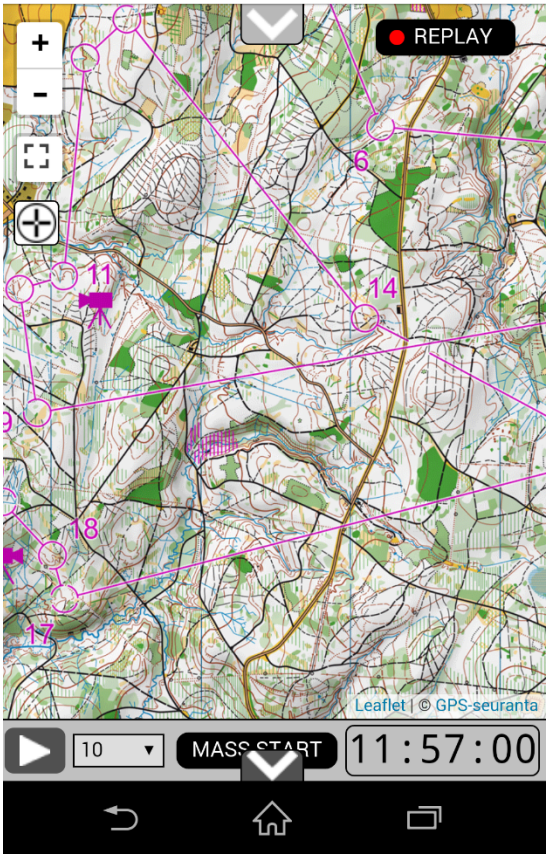
5. Työn tulokset

Projektin valmistumisajankohdaksi oli alunperin asetettu vuoden 2014 toukokuu, mutta tästä jouduttiin luopumaan kun projekti ei edistynyt toivotulla tavalla. Projektin lopulta päätyttyä tammikuussa 2015 oli projektin tavoitteet suurimmaksi osin saatu toteutettua ja projektin jatkokehittäminen siirtyi asiakkaan vastuulle. Myöhästymisistä huolimatta asiakkaalle pystyttiin toimittamaan sovellus, johon asiakas oli suurimmaksi osaksi tyytyväinen ja jota asiakas pystyi viemään eteenpäin tuotantokäyttöön soveltuvaksi.

Projektissa saavutettiin suurin osa asiakkaan kanssa asetetuista vaatimuksista. Kuitenkin projektin edetessä jouduttiin luopumaan tietyistä vaatimuksista. Lisäksi joitain vaatimuksista ei saatu kokonaisuudessaan täytetty. Ainoa vaatimus joka hylättiin täysin, oli vaatimus MVC-arkkitehtuurin käytöstä.

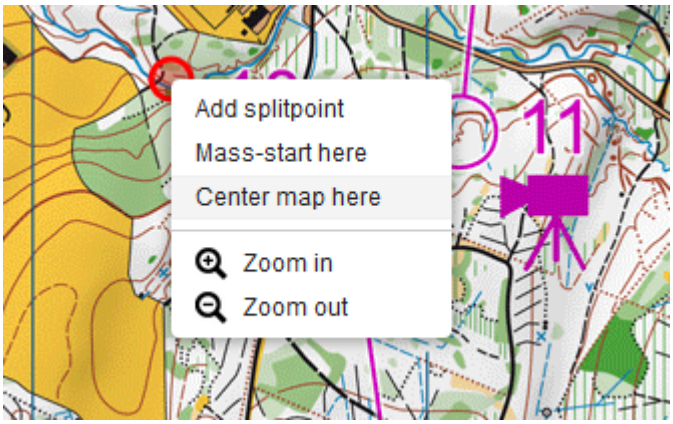
Projektin lopullista versiota pystyi käytännössä käyttämään GPS-datan seuraamiseen reaaliaikaisesti ja uusintana tallennetun koordinaatistodatan avulla. Halutut perustoiminnallisuudet saatiin toteutettua ohjelmistoon. Ohjelman käytettävyys saatiin hiottua asiakkaan haluamaan suuntaan ja lisäksi suurin osa projektin aikana ilmi tulleista bugeista saatiin korjattua. Sovellus saatiin myös toimimaan erittäin kattavalla alusta- ja selaintuella. Käytännössä ongelmia tulee jos käytössä on erittäin vanhentuneet versiot yleisimmin käytössä olevista selaimista. Käytännössä tällaista tilannetta ei kovin suurella todennäköisyydellä tule vastaan tätä sovellusta käyttävien henkilöiden keskuudessa.

Monien vaiheiden jälkeen myös sovelluksen mobiilitilan käyttöliittymä saatiin toteutettua asiakkaan toivomien muutosten mukaiseksi. Käyttöliittymä kokikin etenkin mobiilipuolella monta muodon muutosta koko projektin läpiviennin aikana (kuviot 7. ja 9.).



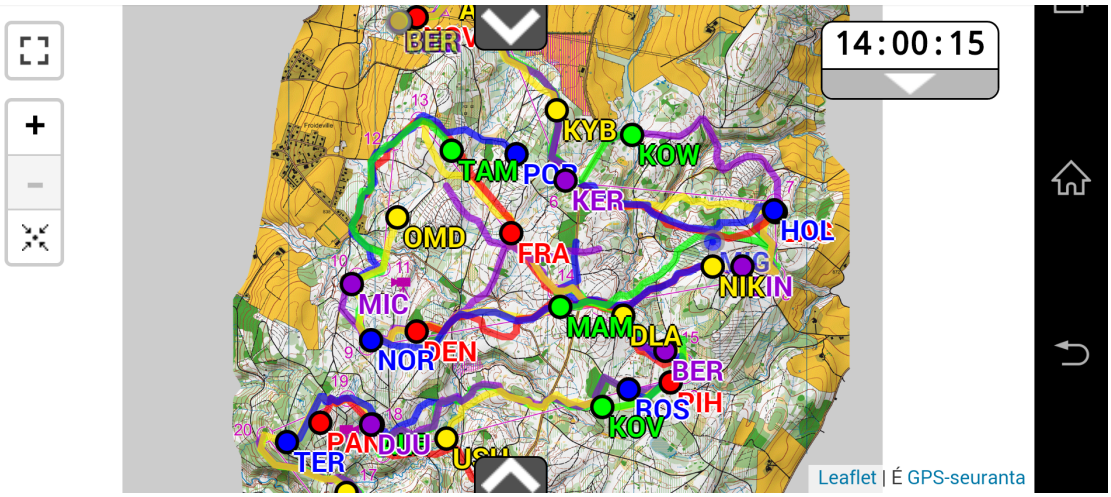
Kuvio 7. Mobiiliversion käyttöliittymän varhainen versio.

Leaflet-kirjasto osoittautui lopulta hyväksi valinnaksi karttadatan käsittelyyn. Vaikka kyseisen kirjaston kehitys ei edistynyt odotetulla tavalla, sen kanssa oli lopulta yllättävän ongelmia odotuksista huolimatta. Käytännössä suurimmat ongelmat kohdistuivat Windows-mobiililaitteisiin, sillä Internet Explorerin mobiiliversio käsitteli tiettyjä näytön kosketustapahtumia eri tavalla muihin selaimiin verrattuna. Nämäkin ongelmat korjaantuivat joko kirjastoon tulleilla bugikorjauksilla tai Windows mobiilikäyttöjärjestelmään tulleiden päivitysten myötä, jotka korjasivat selaimen tukea erilaisille web-standardeille. Leafletin tuki lisäosille osoittautui lopulta erittäin hyödylliseksi. Projektin edetessä tiettyjen asiakkaan toiveiden toteuttaminen oli helppoa, sillä Leafletille oli valmiiksi tarjolla sopivia lisäosia. Esimerkkinä asiakkaan toive hiiren näppäimen painalluksella tai kosketusnäytön pitkällä painalluksella avautuvasta valikosta oli lopulta yksinkertaista toteuttaa käyttämällä ContextMenu -nimistä Leaflet-lisäosaa (kuvio 8.).



Kuvio 8. Leaflet ContextMenu -lisäosa.

Etenkin projektin puolivälin tienoilla mobiiliversion käyttöliittymä oli paikoin erittäin karun näköinen ja käytettävyys ei ollut kovin hyvällä tasolla. Etenkin tietyissä tilanteissa toiminnallisuuden kannalta tärkeät napit saattoivat piiloutua kokonaan muiden elementtien alle (kuvio 9.) tai ne näkyivät sovellusnäkymän ulkopuolella, jolloin käyttäjä ei päässyt käsiksi niihin.



Kuvio 9. Käyttöliittymän mobiilitila.

Suorituskyvyn kannalta sovellus toimii hyvin jopa suurella datamäärällä (paljon kilpailijoita kerralla ja kilpailijan reitit piirretään kokonaisuudessaan kartalle) perinteisillä tietokoneilla. Mobiililaitteilla taas saattaa esiintyä suorituskykyongelmia kilpailuissa, joissa on suuri määrä dataa esitettävänä kerralla kartalle. Tosin nämäkin suorituskyky ongelmat tulee vastaan lähinnä heikompitasoisilla mobiililaitteilla, joissa ei ole käytössä viimeisimpiä ja parhaita järjestelmäpiire-

jä. Projektin kannalta asiakas oli tyytyväinen mobiililaitteidenkin osalta sovelluksen suorituskykyyn testatuilla tapahtumilla.

6. Pohdinta

Suurimmaksi ongelmaksi projektissa muodostui työn alkuun saaminen. Monta viikkoa kului yksinkertaisen käyttöliittymän prototyypin hiomisessa ja mitään kovin konkreettista ei projektin kannalta saatu tehtyä eteenpäin. Pitkään näyttikin siltä, että projekti olisi jäänyt paikoilleen jumittamaan. Lopulta kuitenkin projektin päätyttyä saatiin sovelluksesta valmiiksi sellainen versio, johon asiakas oli tyytyväinen ja jota pystyttiin asiakkaan puolelta kehittämään eteenpäin aina tuotantokäyttöön asti. Onnistuneen projektin kannalta tärkeintä onkin, että saadaan toimitettua sellainen tuote, johon asiakas on tyytyväinen ja joka laadun puolesta on mahdollisimman virheetön. Teknisen toteutuksen ratkaisujen toimivuudesta ja järkevyydestä voidaan aina olla eri mieltä ja käytännössä aina voidaan löytää parempi ratkaisu. Tärkeintä tässä asiassa onkin, että saadaan toimitettua riittävän hyvä pohja, joka täyttää suurimmaksi osin asetetut vaatimukset ja jonka kehittämistä pystytään jatkamaan ilman, että edessä olisi esimerkiksi pian käytettyjen teknologioiden vaihto ja sovelluksen kokoaminen kasaan täysin alusta asti.

Projektia voidaan pitää suurimmaksi osin onnistuneena, vaikka joistakin vaatimuksista jouduttiin luopumaan ja projektissa tavoitteena ollut valmistumisaika liki tuplaantui suunniteltuun nähden. Toisaalta hylätyt vaatimukset olivat sellaisia, jotka eivät sovelluksen käyttämisen kannalta näy juuri mitenkään sovelluksen loppukäyttäjille. Teknisen toteutuksen kannalta etenkin käyttöliittymän mobiili- ja työpöytäversioiden toteutus olisi voinut olla merkittävästi järkevämpi huolellisemmalla suunnittelemisella. Tosin projektin alku kului tehden erittäin huonoksi käynyttä ratkaisua ja aikataulukiireiden takia lähtökohtana jatkolle oli saada aikaiseksi nopeasti uudenlainen toteutus tietyille käyttöliittymän komponenteille, jotta saataisiin aikaiseksi toimiva kokonaisuus eri alustoille käytettäväksi.

IT-alalla työskentelemiseen tulevaisuuden kannalta projekti oli erittäin opettavainen kokemus. Tällä hetkellä erilaiset web-sovellukset ovat erittäin kovassa nousussa. Alustariippuvaisten sovellusten sijaan pyritään tällä hetkellä tarjoamaan web-teknologioilla toteutettuja sovelluksia, jotka toimivat käytännössä

kaikilla laitteilla, jotka tukevat tämänhetkisiä standardeja. Tällä hetkellä projektia voidaan pitää myös työllistämisen kannalta hyödyllisenä, sillä tällä hetkellä IT-alalla on kova kysyntä web-sovelluskehittäjille. Etenkin JavaScript-, HTML5- ja CSS3-osaajia haetaan tällä hetkellä alalla. Projekti tarjosi näin ollen monia uusia asioita opittavaksi, jotka ovat työllistymisen kannalta erittäin oleellisia.

7.1. Jatkokehitys

Vaikka työssä lopulta saavutettiin haluttu lopputulos, jäi työhön myös muutamia selkeitä parannuksen kohteita sovelluksen jatkokehityksen kannalta mietittynä. Esille tulleet ideat keskittyvät pitkälti suorituskyvyn parantamiseen ja sovelluksen latausnopeuden parantamiseen. Osittain muutokset voisivat parantaa sovelluksen ylläpidettävyyttä ja helpottaa uusien ominaisuuksien lisäämistä ja jo olemassa olevien muokkaamista.

Merkittävimmät osa-alueet, joihin parannuksia voisi tehdä, ovat käyttöliittymän toteutus ja ohjelman suorituskykyyn keskittyvät optimoinnit. Käyttöliittymän toteutuksen voisi toisaalta kokonaan refaktoroida. Suorituskykyyn keskittyvät parannukset taas vaatisivat tarkempaa tutustumista sovelluksen lähdekoodiin ja samalla myös tarkempaa perehtymistä JavaScriptiin ja siihen, millaisia optimointeja sen kanssa on mahdollista tehdä järkevästi.

7.2. Käyttöliittymän parannukset

Käyttöliittymän ylläpidettävyyden, sille tehtävien muutoksien ja uusien ominaisuuksien lisäämisen kannalta olisi hyvä pohtia, miten helposti kahden erillisen CSS-tyylitiedoston käytöstä voitaisiin luopua ja pitää kaikki sovelluksen käyttöliittymän tyylimääritykset yhdessä CSS-tiedostossa. Kahden erillisen tyylitiedoston kanssa työskentely on kankeaa ja lisää mahdollisten virheiden määrää, sillä muutokset tulisi muistaa tehdä kumpaankin tiedostoon erikseen. Tosin samanlainen ongelma olisi luultavasti myös yhden CSS-tiedoston käytössä, että muutokset pitäisi muistaa tehdä ”kahteen kertaan”, mutta yhden tiedoston kohdalla on helpompi pysyä mukana sen suhteen, mitä muutoksia tulisi tehdä minnekin.

Nämä muutokset vaatisivat hyvinkin suurta refaktorointia CSS-tiedostojen suhteen ja pieniä muutoksia nykyiseen HTML-sivuun, sillä tällä hetkellä tietyt käyttöliittymäelementit on lisätty kahteen kertaan sivun lähdekoodiin eri luokka- ja ID-ominaisuuksia käyttäen, että halutut elementit on saatu toimimaan eri versioiden välillä oikein. Näin koodista saataisiin helpommin luettavaa, helpommin ylläpidettävää ja käyttöliittymän lisäykset olisi helpompi toteuttaa. Li-

säksi näillä muutoksilla olisi myös pientä vaikutusta sovelluksen latausnopeuksiin. Koodista saataisiin tiiviimpää ja ennen kaikkea ladattavien tiedostojen määrää saataisiin hieman laskettua. Useiden erillisten tiedostojen lataus vaikuttaa juuri sovelluksen aukeamisen nopeuteen, joka näkyy eteenkin hitaampia yhteyksiä käytettäessä.

Suoraan käyttöliittymän ulkoasuun liittyen nykyinen käytössä oleva värimaailma ei ehkä ole kaikkein paras mahdollinen, ja siinä käytössä olevat komponentit olisi mahdollista muuttaa näyttävämmän näköisemmiksi. Käytännössä tämä vaatisi apua henkilöltä, joka on enemmän perillä käyttöliittymän ulkoasun suunnittelemisesta. Ulkoasun viilauksilla tosin saavutettaisiin vieläkin ammattimaisemman näköinen sovellus ja luotaisiin käyttäjille vieläkin laadukkaampi mielikuva sovelluksesta. Tällä voidaan houkutella enemmän potentiaalisia asiakkaita sovelluksen pariin, sillä käyttöliittymän ulkoasu on yksi merkittävä tekijä, jonka perusteella ihmiset tekevät lopulta valintoja, minkä sovelluksen käyttäjiksi he lopulta päätyvät.

7.3. Optimointi ja suorituskyvyn parannukset

Mobiililaitteiden kannalta sovelluksessa olevan JavaScript-koodin optimointi olisi kannattavaa. Muutoksilla saavutettaisiin ehkä jopa merkittävääkin sovelluksen latausnopeuden muutosta ja myös itse sovelluksen käynnistymisaikaa saataisiin nopeutettua heikompitehoisilla laitteilla. Käynnistymisajalla tarkoitetaan tässä yhteydessä sitä, kuinka kauan aikaa kuluu itse kartan ja siihen liittyvän datan hakemiseen, käsittelemiseen ja esittämiseen, ennen kuin itse sovellus on kokonaan käytettävissä.

Yksi optimoinnin kohde olisi useasti toistuvien samankaltaisen funktioiden niputtaminen yhdeksi funktioksi, joka ottaa vastaan useampia parametreja. Esimerkiksi käyttöliittymän tapahtumien hallinnan kohdalla on lähdekoodissa useita samankaltaisia funktioita, joita voisi yrittää saada muutettua toimimaan yhden suuremman funktiokokonaisuuden alla annettavien parametrien mukaan.

Toinen kohde olisi koodin optimointi käytännön suorituskyvyn kannalta. Etenkin muistinkäytön ja sovelluksen sulavan ajamisen kannalta olisi hyvä löytää mahdollisia parannuksen kohteita etenkin mobiililaitteita ajatellen. Muistinkäytön kannalta yksi helppo muutos olisi turhien globaalien muuttujien poistaminen ja niiden siirtäminen funktioiden sisälle, jossa muuttuja vasta luodaan kun funktiota kutsutaan ja samalla poistetaan muistista, kun funktio on suoritettu. Näin muistia ei pidetä kokoajan turhaan varattuna. Muutamia tällaisia muutoksia tehtiin jo projektin aikana lähdekoodiin.

Lähteet

25 Useful JavaScript Libraries And Tools for Creating Interactive Maps. 2013. InstantShift. Viitattu 20.11.2014. http://www.instantshift.com/2013/08/26/useful-javascript-libraries-for-maps/.

50 JavaScript Libraries and Plugins for Maps. N.d. TechSlides. Viitattu 24.11.2014. http://techslides.com/50-javascript-libraries-and-plugins-for-maps.

Arkkitehtuuri ja MVC. N.d. MVC-arkkitehtuuria käsittelevä kirjoitus. Viitattu 26.01.2015. https://advancedkittenry.github.io/koodaaminen/arkkitehtuuri/index.html.

Backbone.js. N.d. Backbone-kirjaston kotisivut. Viitattu 28.11.2014. http://backbonejs.org/.

Backbone.js. 2014. Wikipedia. Viitattu 28.12.2014. http://en.wikipedia.org/wiki/Backbone.js.

Canvas – HTML. 2013. MDN Canvas pääsivu. Viitattu 16.11.2014. https://developer.mozilla.org/fi/docs/HTML/Canvas.

Canvas element. N.d. Wikipedia. Viitattu 04.12.2014. http://en.wikipedia.org/wiki/Canvas_element.

Compare Projects. N.d. Open Hub. Viitattu 23.01.2015. https://www.openhub.net/p/compare?project_0=OpenLayers&project_1=Leaflet.

Google API Tutorial. N.d. Viitattu 09.01.2015. http://www.w3schools.com/googleapi.

Google Developers. N.d. Viitattu 27.11.2014. https://developers.google.com/maps/-

Google Maps. N.d. Wikipedia. Viitattu 09.01.2015. http://en.wikipedia.org/wiki/Google_Maps.

How do various JavaScript mapping libraries compare? N.d. - Geographic Information Systems Stack Exchange. Viitattu 10.11.2014. http://gis.stackexchange.com/questions/8032/how-do-various-javascript-mapping-libraries-compare.

jQuery. N.d.a. jQuery-kirjaston kotisivut. Viitattu 06.12.2014. https://jquery.com/.

jQuery. N.d.b. Wikipedia. Viitattu 05.12.2014. http://en.wikipedia.org/wiki/JQuery.

jQuery Foundation. N.d. jQuery-säätiön kotisivut. Viitattu 13.12.2014. https://jquery.org/.

jQuery Color Picker colpick. N.d. jQuery Color Picker colpick-kirjaston kotisivut. Viitattu 11.12.2014. http://colpick.com/plugin.

jQuery UI. N.d. jQuery UI-kirjaston kotisivut. Viitattu 07.12.2014. https://jqueryui.com/.

Leaflet - a JavaScript library for mobile-friendly maps. N.d. Leaflet-kirjaston kotisivut. Viitattu 12.12.2014. http://leafletjs.com/.

Leaflet (software). N.d. Wikipedia. Viitattu 24.11.2014. http://en.wikipedia.org/wiki/Leaflet_%28software%29.

MVC Architecture. N.d. MVC-arkkitehtuuria käsittelevä kirjoitus. Viitattu 26.11.2014. http://www.tutorialspoint.com/struts_2/basic_mvc_architecture.htm.

MVC-arkkitehtuuri. N.d. Wikipedia. Viitattu 20.11.2014. http://fi.wikipedia.org/wiki/MVC-arkkitehtuuri.

MVC- Malli, peruskauraa frameworkkien käyttäjille. 2010. MVC-mallia käsittelevä blogikirjoitus. Viitattu 10.12.2014. https://vahtolam.wordpress.com/2010/09/23/mvc-malli-peruskauraa-frameworkkien-kayttajille/.

OpenLayers. N.d. Wikipedia. Viitattu 27.11.2014. http://en.wikipedia.org/wiki/OpenLayers.

OpenLayers 2. N.d. OpenLayers 2-kirjaston kotisivut. Viitattu 23.11.2014. http://openlayers.org/two/.

OpenLayers 3 – Welcome. N.d. OpenLayers 3-kirjaston kotisivut. Viitattu 25.11.2014. http://openlayers.org/.

Spin.js. N.d. Spin-kirjaston kotisivut. Viitattu 04.01.2015. http://fgnass.github.io/spin.js/.

Testing web map APIs - Google vs OpenLayers vs Leaflet. 2014. Kartta rajapintoja käsittelevä blogikirjoitus. Viitattu 10.01.2015. http://robinlovelace.net/software/2014/03/05/webmap-test.html.

Web Hypertext Application Technology Working Group. N.d. WHATWG-järjestön kotisivut. Viitattu 20.01.2015. https://whatwg.org/.

WHATWG. N.d. Wikipedia. Viitattu 21.01.2015. http://en.wikipedia.org/wiki/WHATWG.